Two Pointers Cheatsheet

Use this algorithm when:

- We are looking for the “best” continuous range which is valid
    - Define valid, and define what makes one range better than another range
- If we remove an element from a valid range, the range will still be valid
- If we add an element to a range:
    - The range may be invalid
    - If the new range is not invalid, adding the element will either keep the range the same or make the range “better” (will not make the range worse)

Steps to using the algorithm:

Thinking about the problem:

1) Decide how you want to represent a range
    a) Usually integers or a set
2) Figure out how do the add and remove functions
3) Figure out how to determine if a range is valid
4) Figure out how to update the answer with the current range

Implementation format:

- Fill in these 4 things:
    - Add function
    - Remove function
    - Determining if the range is valid
    - Updating the answer

```
int lp = 0;
for (int rp=0; rp<N; rp++) {
      add (rp);
      while (range is not valid) {
            remove (lp);
            lp++;
      }
      update answer;
}
```

How to calculate the runtime:

O(N * (runtime of the add and remove functions))